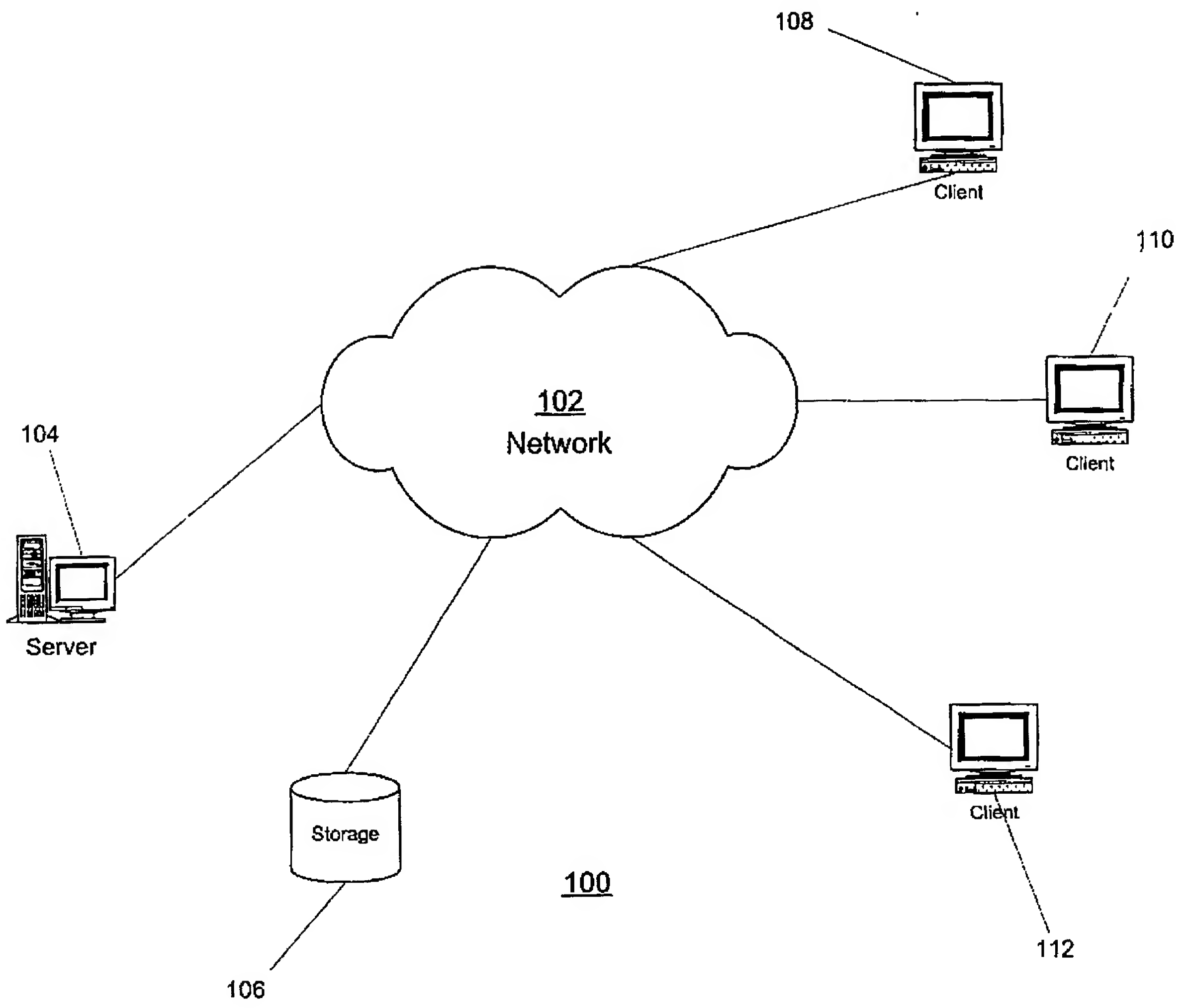
ET420197807US

Figure 1

Koved et al.
AUS920010941US1
Method and Apparatus for Type
Independent Permission Based
Access Control
Page 1 of 13



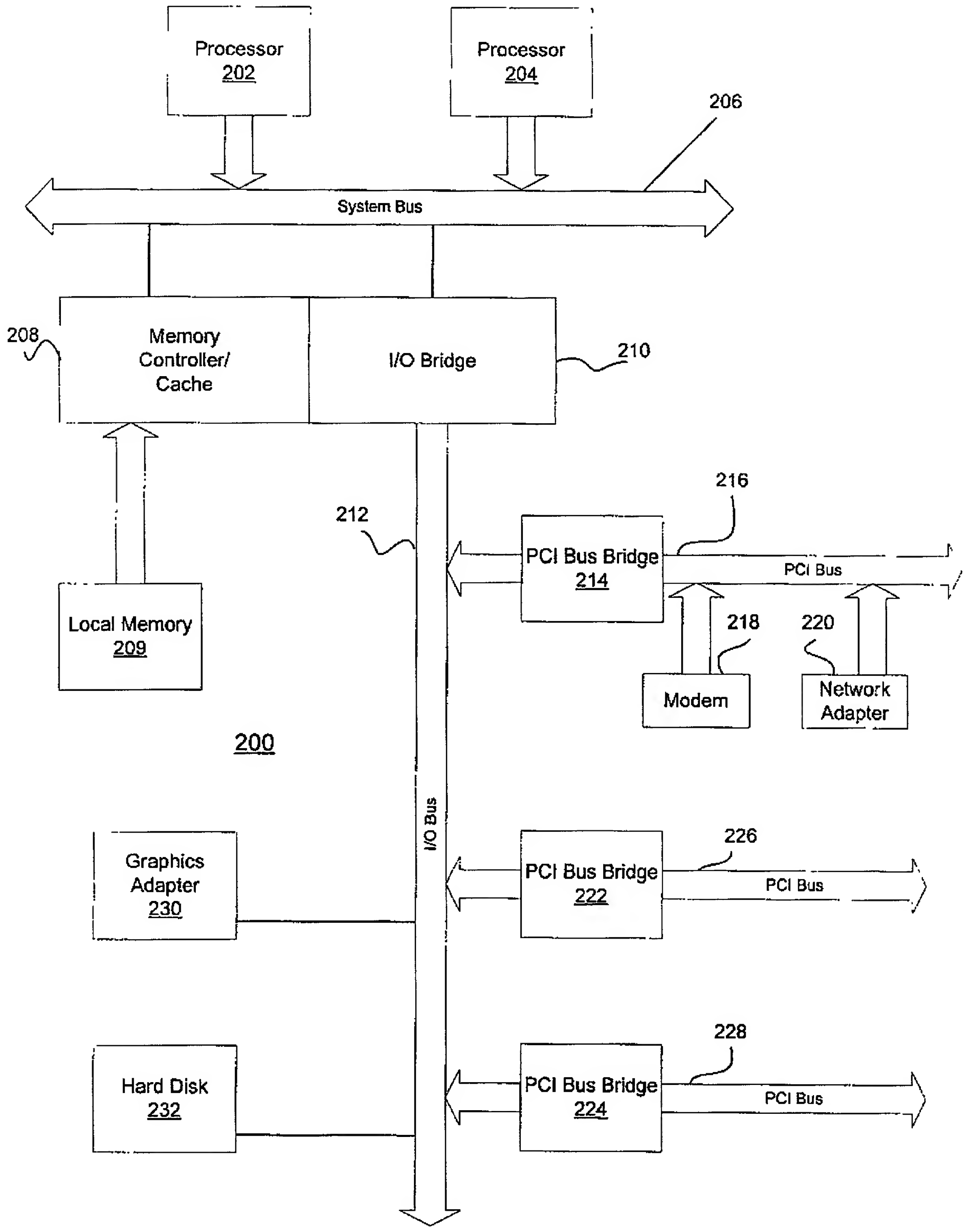


Figure 2

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control
Page 2 of 13

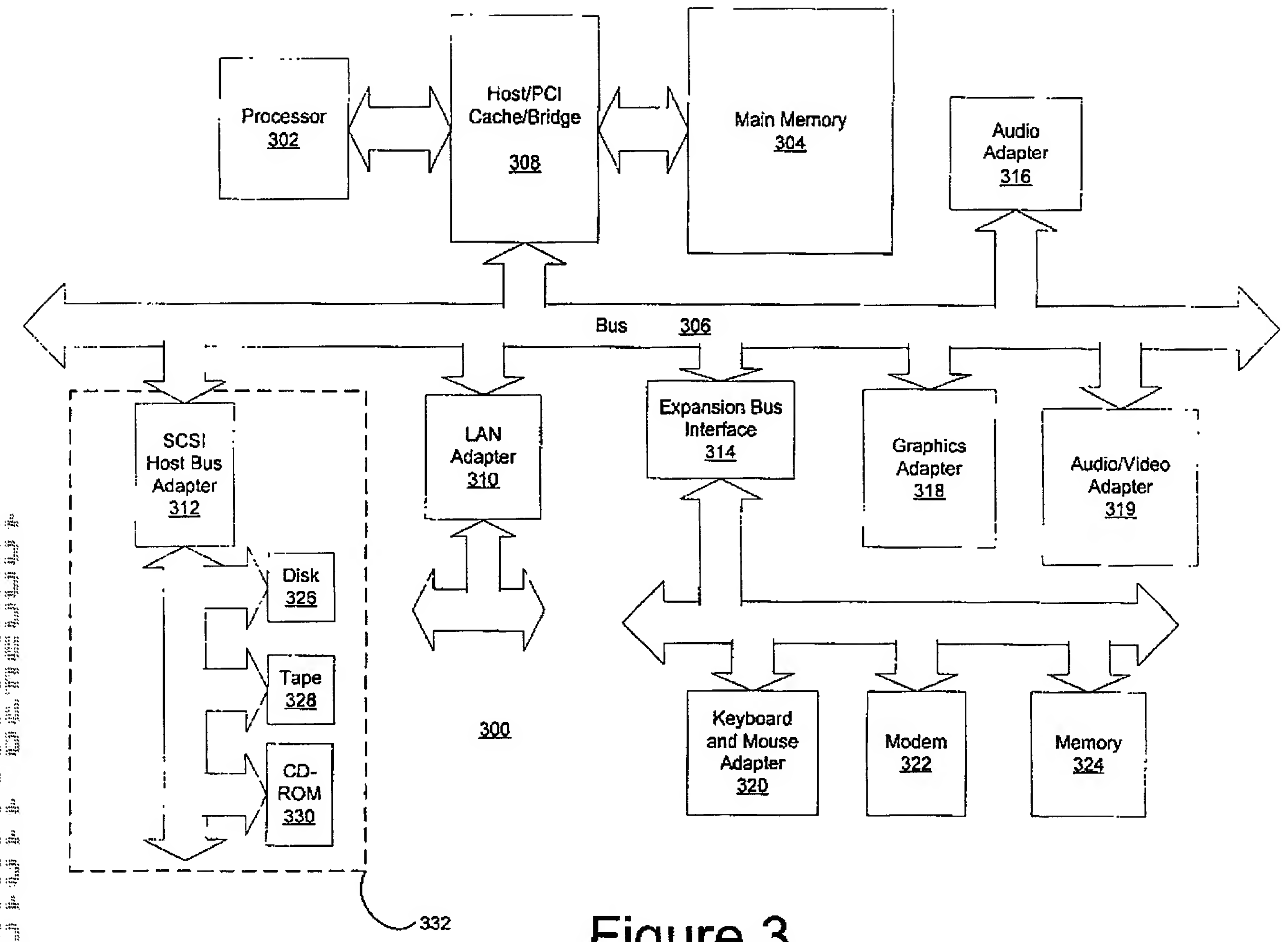


Figure 3

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control
Page 3 of 13

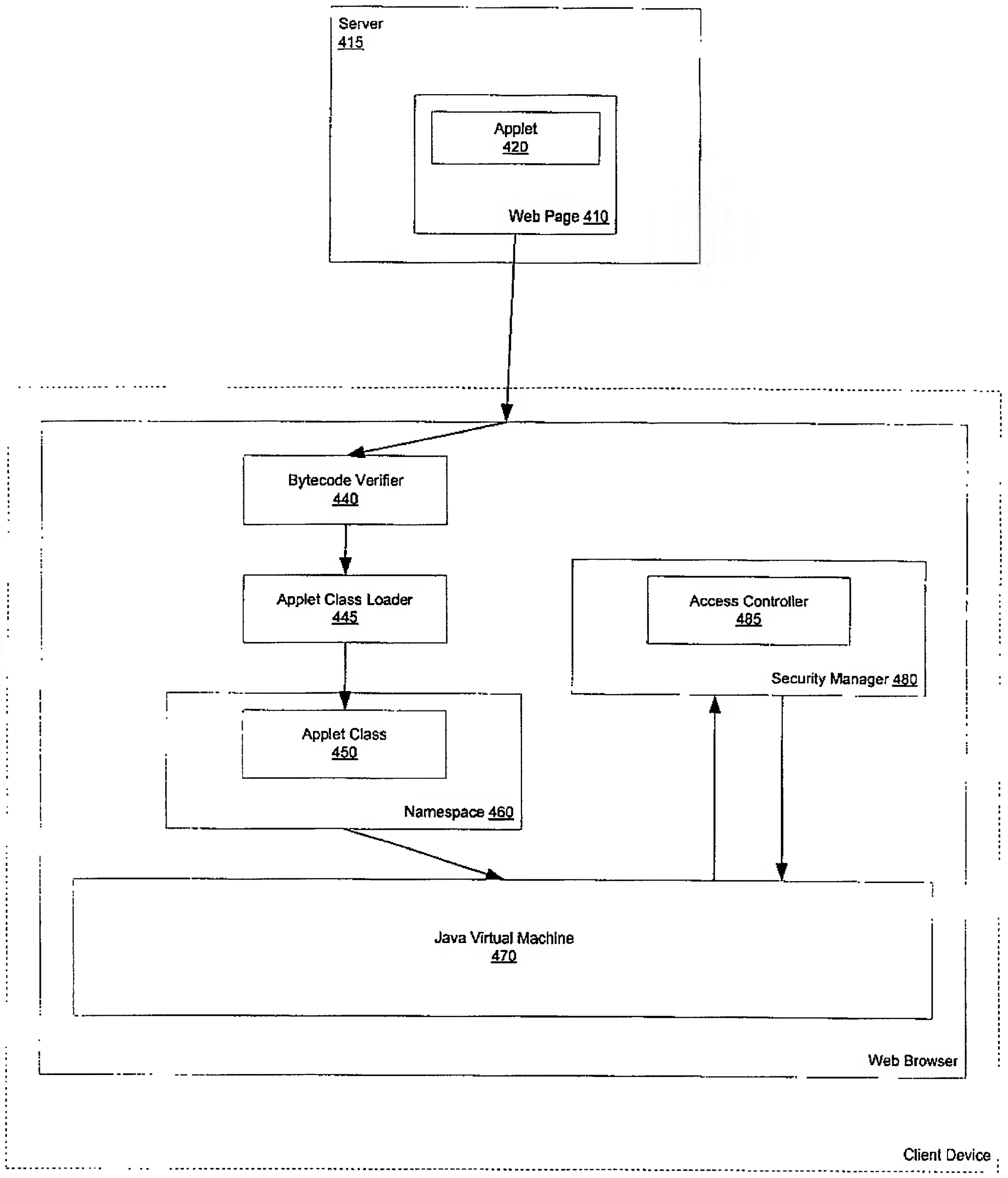


Figure 4

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control
Page 4 of 13

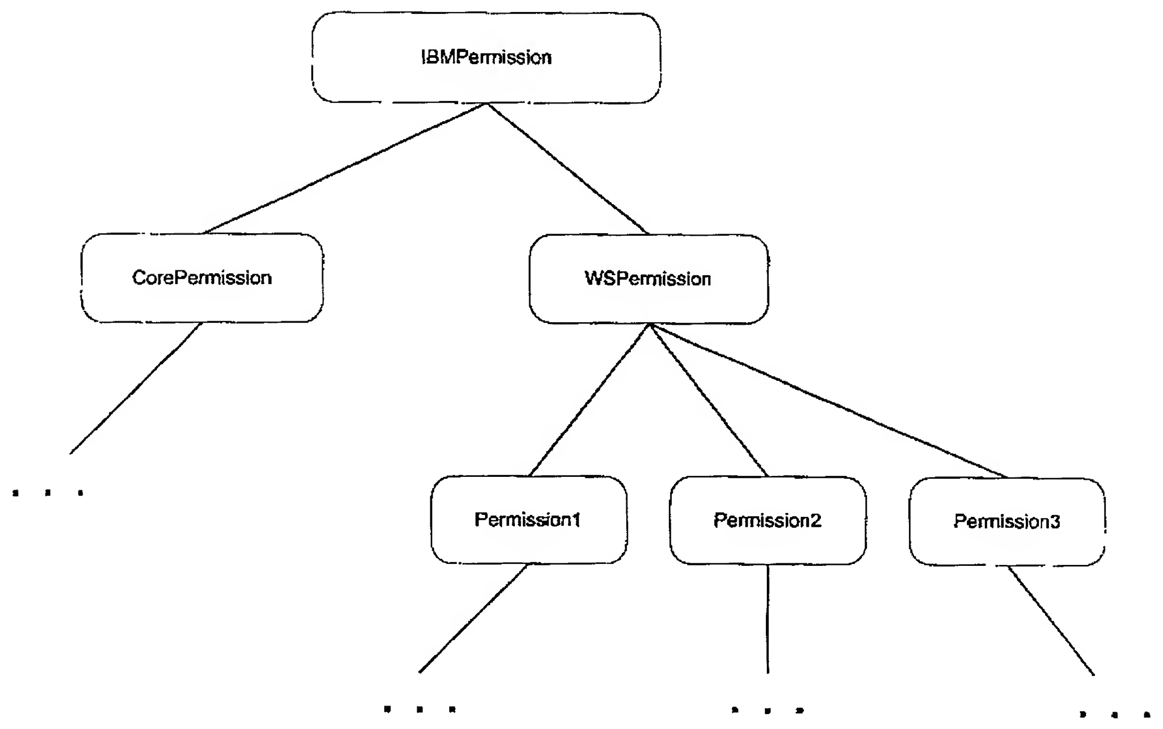

IBMPermission
CorePermission
WSPermission
Permission1
Permission2
Permission3
. . .
. . .
. . .
. . .

Figure 5

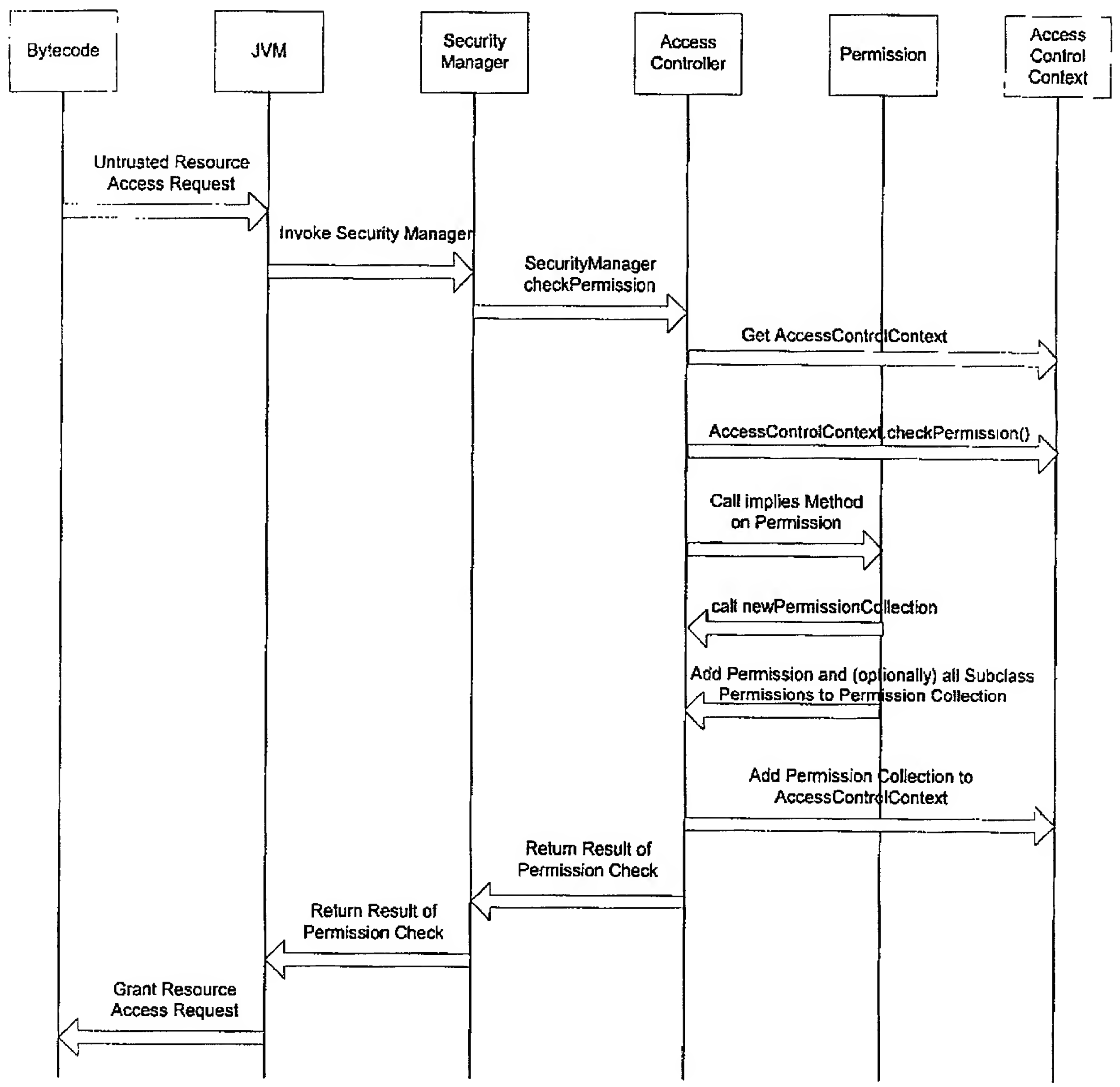

Bytecode
JVM
Security Manager
Access Controller
Permission
Access Control Context
Untrusted Resource Access Request
Invoke Security Manager
SecurityManager checkPermission
Get AccessControlContext
AccessControlContext.checkPermission()
Call implies Method on Permission
call newPermissionCollection
Add Permission and (optionally) all Subclass Permissions to Permission Collection
Add Permission Collection to AccessControlContext
Return Result of Permission Check
Return Result of Permission Check
Grant Resource Access Request

Figure 6

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control
Page 6 of 12

```
import java.security.BasicPermission;
import java.security.Permission;
import java.security.PermissionCollection;
import java.util.Hashtable;
import java.util.Enumeration;

public class IBMPermission extends BasicPermission
{
        public IBMPermission()
        {
                super("");
                System.out.println("Constructor IBMPermission() called");
        }
        public IBMPermission(String target)
        {
                super(target);
                System.out.println("Constructor IBMPermission(target) called");
        }

        public IBMPermission(String target, String actions)
        {
                super(target, actions);
                System.out.println("Constructor IBMPermission(target, actions) called");
        }
        public boolean implies(Permission perm)
        {
                System.out.println("IBMPermission.implies() called");

                if (perm instanceof IBMPermission)
                        return true;
                return false;
        }
        public PermissionCollection newPermissionCollection()
    {
                return new IBMPermissionCollection();
    }
}
```

Figure 7A

```
final class IBMPermissionCollection extends PermissionCollection
        implements java.io.Serializable
{
    private Hashtable permissions;

    public IBMPermissionCollection()
    {
                permissions = new Hashtable();
    }

    public void add(Permission permission)
    {
                if (! (permission instanceof IBMPermission))
                    throw new IllegalArgumentException("Invalid Permission: " +
                                                                        permission);

                IBMPermission ibmp = (IBMPermission) permission;
                permissions.put(ibmp.getName(), permission);
    }

    public boolean implies(Permission permission)
    {
                if (! (permission instanceof IBMPermission))
                        return false;

                System.out.println("permission instanceof IBMPermission == true");

                IBMPermission ibmp = (IBMPermission) permission;
                String permName = ibmp.getName();
                Permission x = (Permission) permissions.get(permName);

                if (x != null)
                {
                    System.out.println("We have a direct hit! " + x.getName());
                    return x.implies(permission);
                }

                Enumeration permEnum = permissions.elements();

                while (permEnum.hasMoreElements())
                {
                        x = (IBMPermission) permEnum.nextElement();
                        System.out.println(x.getName());

                        if (x.implies(permission))
                                return true;
                }

                return false;
        }

    public Enumeration elements()
    {
                return permissions.elements();
    }
}
```

Figure 7B

```
import java.security.PermissionCollection;
import java.security.AccessController;
import java.security.AccessControlContext;
import java.security.AccessControlException;

public class WSPermission extends IBMPermission
{
        public WSPermission(String target)
        {
                super(target);
                System.out.println("Constructor WSPermission(target) called");
        }

        public WSPermission(String target, String actions)
        {
                super(target, actions);
                System.out.println("Constructor WSPermission(target, actions) called");
        }

        public WSPermission()
        {
                super("");
                System.out.println("Constructor WSPermission() called");
        }

    /**
     * Returns a new IBMPermissionCollection object for storing IBMPermission
     * objects.
     * <p>
     * An IBMPermissionCollection stores a collection of
     * IBMPermission permissions.
     * <p>
     * IBMPermission objects must be stored in a manner that allows them
     * to be inserted in any order, but that also enables the
     * PermissionCollection <code>implies</code> method
     * to be implemented in an efficient (and consistent) manner.
     *
     * @return a new IBMPermissionCollection object suitable for
     *          storing IBMPermission's.
     */
        public PermissionCollection newPermissionCollection()
    {
                System.out.println("newPermissionCollection() was called");
                IBMPermissionCollection ibmPC = new IBMPermissionCollection();

                // the code here checks if an IBMPermissionCollection has been granted.
                // If yes, then the PermissionCollection returned by this
                // method should contain a WSPermission.

                AccessControlContext acc = AccessController.getContext();

                try
                {
                        acc.checkPermission(new IBMPermission("PermissionTest"));
                        ibmPC.add(new WSPermission("PermissionTest"));
                }
                catch (AccessControlException ace)
                {
                        System.out.println("IBMPermission WAS NOT GRANTED");
                }
                return ibmPC;
    }
}
```

Figure 7C

Koved et al.
AUS920010941US1
Method and Apparatus for Type
Independent Permission Based
Access Control

```
import java.io.*;

public class PermissionTest
{
      public static void main(String args[])
      {
            try
            {
                  SecurityManager sm = System.getSecurityManager();

                  if (sm != null)
                  {
                        System.out.println("SecurityManager is checking for " +
                                                      "WSPermission");

                        sm.checkPermission(new WSPermission("PermissionTest"));
                  }

                  System.out.println("WSPermission was granted. " +
                                                "Permission testing
worked.\n\n\n");

                  File inputFile = new File("C:\\winzip.log");
                  FileInputStream fis = new FileInputStream(inputFile);
                  InputStreamReader isr = new InputStreamReader(fis);
                  BufferedReader br = new BufferedReader(isr);

                  String lineRead;
                  while ((lineRead = br.readLine()) != null)
                        System.out.println(lineRead);
            }

      catch(Exception e)
      {
                  e.printStackTrace();
      }
      }
}
```

Figure 8

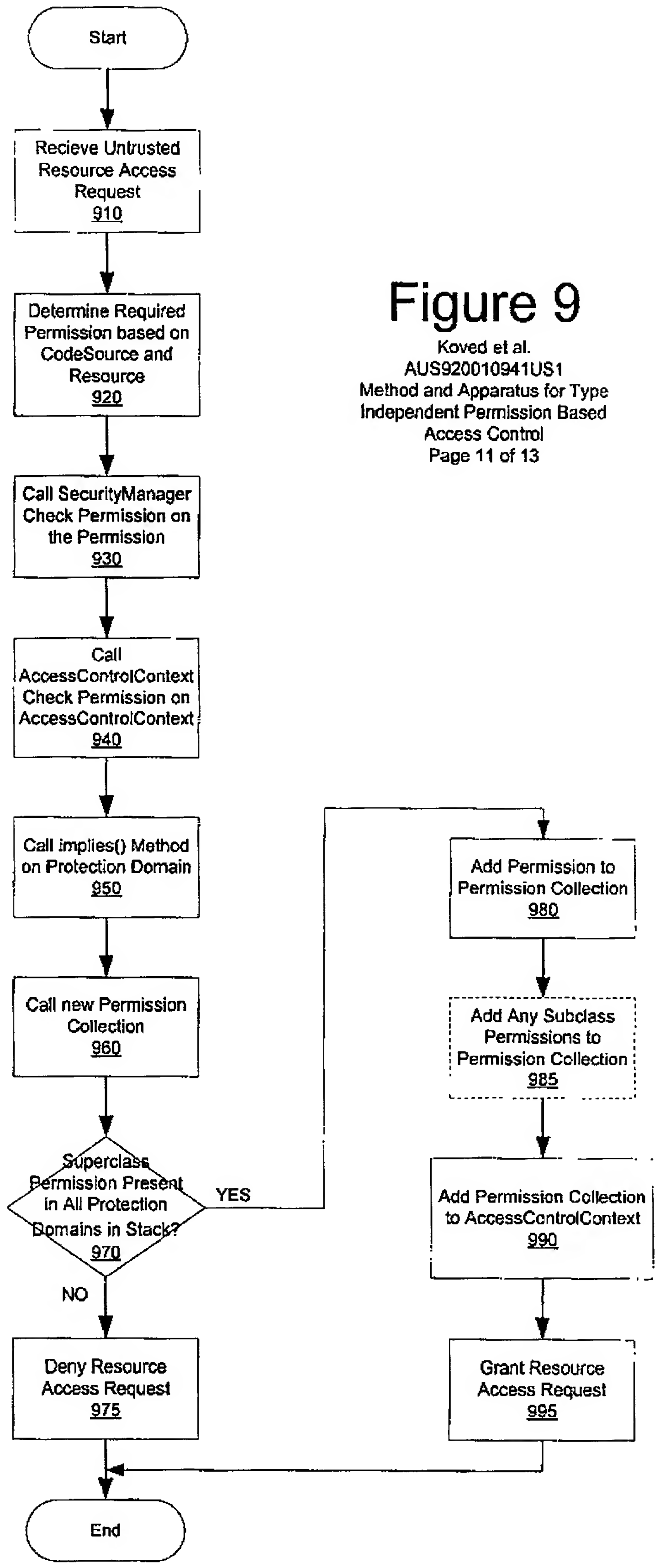

Start
Recieve Untrusted Resource Access Request 910
Determine Required Permission based on CodeSource and Resource 920
Call SecurityManager Check Permission on the Permission 930
Call AccessControlContext Check Permission on AccessControlContext 940
Call implies() Method on Protection Domain 950
Call new Permission Collection 960
Superclass Permission Present in All Protection Domains in Stack? 970
YES
NO
Add Permission to Permission Collection 980
Add Any Subclass Permissions to Permission Collection 985
Add Permission Collection to AccessControlContext 990
Deny Resource Access Request 975
Grant Resource Access Request 995
End

Figure 9

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control

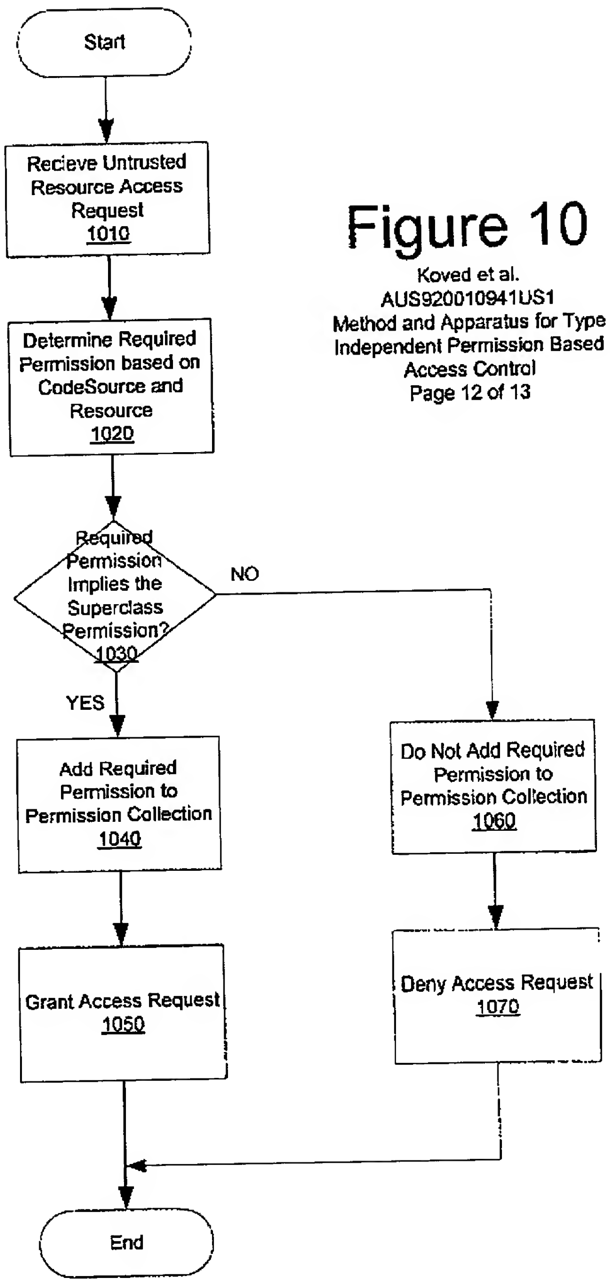

Start
Recieve Untrusted Resource Access Request 1010
Determine Required Permission based on CodeSource and Resource 1020
Required Permission Implies the Superclass Permission? 1030
NO
YES
Add Required Permission to Permission Collection 1040
Grant Access Request 1050
Do Not Add Required Permission to Permission Collection 1060
Deny Access Request 1070
End

Figure 10

Koved et al.
AUS920010941US1
Method and Apparatus for Type Independent Permission Based Access Control
Page 12 of 13

```
package sun.security.provider;

import java.security.PermissionCollection;
import java.security.CodeSource;
import IBMPermission;
import WSPermission;

public class MarcoPolicy extends PolicyFile
{
      public PermissionCollection getPermissions(CodeSource codesource)
      {
            PermissionCollection pc = super.getPermissions(codesource);

            if (pc == null)
                  return null;

            if (pc.implies(new IBMPermission("PermissionTest")))
                  pc.add(new WSPermission("PermissionTest"));

            return pc;
      }
}
```

Figure 11